View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Chinese Journal of Aeronautics 24 (2011) 202-209

Contents lists available at ScienceDirect

Chinese Journal of Aeronautics

journal homepage: www.elsevier.com/locate/cja

Update Chain-based Approach for Checking Route Oscillation of BGP

ZHANG Jun[a,b,*], HU Ziying[a,b], ZHANG Tao[a,b]

[a]*School of Electronics and Information Engineering, Beihang University, Beijing 100191, China*

[b]*National Key Laboratory of CNS/ATM, Beijing 100191,China*

Received 25 June 2010; revised 11 October 2010; accepted 21 December 2010

Abstract

This paper presents a novel approach for checking route oscillation of border gateway protocol (BGP) quickly, by which the privacy of routing policies of autonomous system (AS) can be respected. Firstly, route update chain tag (RUCT) is constructed to track the forwarding of update report, and local routing library is made to record the changing history of update report. Then route oscillation can be identified by analyzing correlative state of RUCT and local routing library. The characteristic of this approach is that an arbitrary AS can check route oscillation alone only by sharing its network ID, which greatly respects the privacy of routing policies for each AS. This paper shows that both looping in RUCT and consecutive positive-negative report in local record are sufficient conditions for route oscillation. Comparative experiments demonstrate the availability and efficiency of the proposed approach.

Keywords: routing protocols; gateways; border gateway protocol; autonomous systems; interdomain routing; oscillations

1. Introduction

The border gateway protocol (BGP)[1] is a path vector protocol for exchanging routing information between autonomous systems (ASs), which is currently the only inter-domain routing protocol applied to the Internet. BGP allows each AS to independently formulate and regulate its routing policies for reasons of local rationality, and override the rationality of whole network. It is reported that BGP is usually weak in operational and security issue as shown in Ref.[2]. Current research on BGP focuses on exposing and resolving both operational and security concerns. Operational concerns[3-5] deal with convergence delay, routing stability and performance of BGP, while security research[6-8] relates to the integrity, confidentiality and authentication of BGP messages. These two fields of operational issues and security research are inherently connected. Route oscillation, an important factor to impact routing stability, has been the focus of many efforts. BGP route oscillation can significantly degrade the end-to-end network performance and depress the overall efficiency of network.

Broadly speaking, this problem can be addressed either statically or dynamically. A static solution would rely on programs to pre-analyze routing policies and verify that they do not contain policy conflict which will lead to route oscillation. For example, Routing Arbiter project is designed in Ref.[9] to check route oscillation. This project consists of three elements: 1) a common description language for routing policy specification; 2) a distributed registry system that allows providers to publish their policies in this language; and 3) a suite of tools for analyzing the impact of provider routing policies on Internet traffic. However, it is difficult to get detailed routing policies and path information, and research of Ref.[10] indicates that checking for various global convergence conditions is either

*Corresponding author. Tel.: +86-10-82338282.
E-mail address: buaazhangjun@vip.sina.com
Foundation items: National Basic Research Program of China (2011CB707000); Foundation for Innovative Research Groups of the National Natural Science Foundation of China (60921001)

1000-9361 © 2011 Elsevier Ltd. Open access under CC BY-NC-ND license.
doi: 10.1016/S1000-9361(11)60024-1

NP-complete or NP-hard. Considering this problem, Refs.[11]-[12] improve the acquiring mechanism of routing policies and present subnet-relationship-based approaches for detecting policy conflict. The dynamic solution is a certain mechanism to suppress or completely prevent BGP route oscillation at run time by extending BGP protocol. For example, route flap damping (RFD) presented in Ref.[13] is a dynamic approach for preventing route oscillation. RFD includes two important parameters, route penalty and threshold, by which route update will be restrained if penalty value exceeds the upper limit of threshold, or be restarted if penalty value falls back to the lower limit of threshold. Ref.[14] shows that RFD can significantly exacerbate the convergence times of relatively stable routes, then makes some improvements on scheme. However, RFD cannot really eliminate route oscillation which arises from policy conflict. It can only make oscillation run in slow motion[15]. Ref.[16] presents a mechanism to tradeoff policy independence and route stability. This mechanism ensures policy independence as soon as possible and prefers to route stability in need. In addition, it is necessary to obtain enough route information of subnets. Ref.[17] proposes simple path vector protocol (SPVP) and constructs dispute direct graph according to route history which is involved in update report. By SPVP, real-time detection of route oscillation can be conducted, but routing policies of subnets have to be shared and network cost of BGP should be increased.

As shown above, it is the same characteristic for static solution and dynamic solution that each subnet has to share policy information of itself and cooperate to resolve route oscillation problem. In other words, one subnet cannot check route oscillation only by itself alone. Both solutions can ensure the stability of BGP route, but override the privacy of routing policies of subnets. However, BGP is vulnerable in nature[18]. Ref.[19] indicates that clustering network, such as global information grid (GIG), has more reasons to be concerned about BGP security than the commercial Internet. It also denotes that keeping the confidentiality of peering sessions is urgent for BGP security. Therefore, many subnets of GIG or space information network[20] do not widely share their routing policies, or only publish incomplete specifications considering security and privacy. In this case, it is unfeasible for current solutions to check route oscillation in those networks.

Considering the privacy of routing policies, Ref.[21] presents an approach based on dispute direct graph, in which detailed path information is replaced by relative preference. Though this approach respects the privacy of subnets, it is difficult to optimize the threshold for relative preference conflict and it will take a long time to detect route oscillation. Ref.[22] introduces a distributed method to dynamically check BGP route oscillation. According to this approach, each node will create and forward an oscillation flag, called TOKEN, when detecting an oscillation on local path. If the TOKEN comes full circle back to source node, it is affirmed that this route oscillation should occur at source node. Because both local path oscillation and oscillation cycle solely have to be checked, this approach will spend more time in resolving route oscillation too. Moreover, it is necessary to encrypt TOKEN for confidentiality, which is more complex. Ref.[23] proposes a distributed mechanism to eliminate route oscillation. This approach enforces a global preference value on each optional path and prioritizes those paths according to their preference values firstly. The preference value of selected path should be increased by 1 if policy conflict happens. Though this mechanism need not forward any policy information of local node and greatly respects the privacy of routing policies, the ultimate route tree is not optimal, in which many optional paths with higher priority are discarded by error. Ref.[24] offers an approach to check oscillation on source node where routing policies have been modified. But it is not applicable to other nodes except source node, and the path history has to be encrypted too.

This paper presents a novel approach to rapidly check route oscillation on arbitrary subnet. It greatly respects privacy of route policies, and each subnet should only share simple network ID rather than publish detailed path information.

2. Sufficient Condition of Route Oscillation

Let $G=(V, E)$ be a directed graph to show overall network in which each subnet is acted as a node, where V is the set of all nodes and E the set of all edges' connecting nodes. The direction of edge is decided as follows.

Definition 1 For $u,v\in V$, node v is neighbor of u, and $e_{uv}\in E$. If path information advertised by v is accepted by Adj-RIB-In of u, the direction of edge e_{uv} is $v\rightarrow u$, called transmission edge. Moreover, e_{uv} is called input transmission edge for u and output transmission edge for v.

Definition 2 If path information advertised by both v and u is refused by each other, then $e_{uv}=\varepsilon$, called empty edge.

Definition 3 s_t^v denotes the optimal path for v at time t. $\lambda^v(p_1^v)$ represents priority of p_1^v, which is a feasible path of v.

For the sake of simplicity, this paper considers only essential characteristic of BGP, and assume that 1) overlapping route and aggregation are ignored, 2) MED and ORGIN attributes are ignored, 3) there is only one connection between ASs, 4) the matter about interior border gateway protocol (IBGP) is ignored, 5) ID of each AS is unique.

Given $G=(V, E)$ in which routing policies are stable at each node initially, $u,v\in V$, $e_{uv}\in E$. At time t, the optimal path of v is changed for some reasons. Then v advertises an update report $r^v=(\uparrow p_1^v, \downarrow p_2^v)$ to neighbor

nodes along $e_{uv}=(u,v)$, where p_1^v is the current optimal path that will be selected by v and p_2^v is the former optimal path that should be withdrawn by v. Suppose that both p_1^v and p_2^v should be accepted by Adj-RIB-In of u (or else their priority is regarded as 0 at u). According to local policies of u, there are four cases of response output from u after decision-making at time $t+1$, shown in Table 1.

Table 1 Response output of u after receiving r^v

s_t^u	s_{t+1}^u	
	p^u	$(uv)p_1^v$
p^u	—	$r^u=(\uparrow (uv)p_1^v, \downarrow p^u)$
$(uv)p_2^v$	$r^u=(\uparrow p^u, \downarrow (uv)p_2^v)$	$r^u=(\uparrow (uv)p_1^v, \downarrow (uv)p_2^v)$

In Table 1, p^u is the path with the highest priority among all feasible paths which do not include node v in Adj-RIB-In of u, contrary to $(uv)p^v$. The four cases are presented as follows:

(1) If $s_t^u = p^u$ and $s_{t+1}^u = p^u$, which means $\lambda^u(p^u) > \lambda^u((uv)p_1^v)$ and $\lambda^u(p^u) > \lambda^u((uv)p_2^v)$, the current optimal path of u is not affected by the change of optimal path at node v. Then u does not advertise any update report to neighbor nodes after receiving r^v.

(2) If $s_t^u = p^u$ and $s_{t+1}^u = (uv)p_1^v$, which means $\lambda^u((uv)p_1^v) > \lambda^u(p^u) > \lambda^u((uv)p_2^v)$, the optimal path of u should be changed to $(uv)p_1^v$ after receiving r^v. Then u will advertise an update report $r^u=(\uparrow (uv)p_1^v, \downarrow p^u)$ after decision-making (see Fig.1(a)).

(3) If $s_t^u = (uv)p_2^v$ and $s_{t+1}^u = p^u$, which means $\lambda^u((uv)p_2^v) > \lambda^u(p^u) > \lambda^u((uv)p_1^v)$, the optimal path of u should be changed to p^u after receiving r^v. Then u will advertise an update report $r^u=(\uparrow p^u, \downarrow (uv)p_2^v)$ after decision-making (see Fig.1(b)).

(4) If $s_t^u = (uv)p_2^v$ and $s_{t+1}^u = (uv)p_1^v$, which means $\lambda^u((uv)p_1^v) > \lambda^u(p^u)$ and $\lambda^u((uv)p_2^v) > \lambda^u(p^u)$, the optimal path of u should be changed to $(uv)p_1^v$ after receiving r^v. Then u will advertise an update report $r^u=(\uparrow (uv)p_1^v, \downarrow (uv)p_2^v)$ after decision-making (see Fig.1(c)).

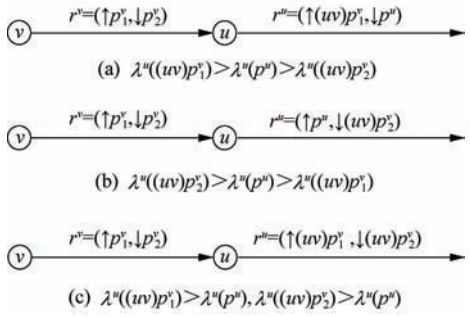

Fig.1 Response output of u after receiving r^v.

Definition 4 Provided that both p_1^v and p_2^v are feasible paths of v, and $r^v=(\uparrow p_1^v, \downarrow p_2^v)$ is a route update report of v, then the other update report $(\uparrow p_2^v, \downarrow p_1^v)$ is negative report of r^v, denoted as $\overline{r^v}$.

Theorem 1 For $G=(V, E)$, $u,v \in V$, $e_{uv} \in E$. If at time t the route update report r^v results in that u advertises r^u, then at next time $t+1$ the report $\overline{r^v}$ should result in the fact that u advertises $\overline{r^u}$.

Proof At time t, v advertises an update report $r^v=(\uparrow p_1^v, \downarrow p_2^v)$ to u, where p_1^v is the current optimal path that will be selected by v and p_2^v the former optimal path that should be withdrawn by v. Suppose both p_1^v and p_2^v should be accepted by Adj-RIB-In of u (or else their priority is regarded as 0 at u). According to local policies of u, there are four cases of response output at time $t+1$.

(1) If $\lambda^u(p^u) > \lambda^u((uv)p_1^v)$ and $\lambda^u(p^u) > \lambda^u((uv)p_2^v)$, the current optimal path of u is not affected by the changing of optimal path at node v. As shown in Fig.2(a), u does not advertise any update report at time t and $t+1$. It is not in conflict with this theorem.

(2) If $\lambda^u((uv)p_1^v) > \lambda^u(p^u) > \lambda^u((uv)p_2^v)$, the current optimal path of u is $s_t^u = p^u$ at time t. After receiving $r^v=(\uparrow p_1^v, \downarrow p_2^v)$, u should update optimal path to $(uv)p_1^v$ and advertise $r^u=(\uparrow (uv)p_1^v, \downarrow p^u)$ to neighbor nodes. At time $t+1$, the current optimal path of u is $s_{t+1}^u = (uv)p_1^v$ and v advertises $\overline{r^v}=(\uparrow p_2^v, \downarrow p_1^v)$ again. As shown in Fig.2(b), u should update optimal path to p^u after decision-making and advertise $\overline{r^u}=(\uparrow p^u, \downarrow (uv)p_1^v)$ to neighbor node again. The theorem is true.

(3) If $\lambda^u((uv)p_2^v) > \lambda^u(p^u) > \lambda^u((uv)p_1^v)$, the current optimal path of u is $s_t^u = (uv)p_2^v$ at time t. After receiving $r^v=(\uparrow p_1^v, \downarrow p_2^v)$, u should update optimal path to p^u and advertise $r^u=(\uparrow p^u, \downarrow (uv)p_2^v)$ to neighbor nodes. At time $t+1$, the current optimal path of u is $s_{t+1}^u = p^u$ and v advertises $\overline{r^v}=(\uparrow p_2^v, \downarrow p_1^v)$ again. As shown in Fig.2(c), u should update optimal path to $(uv)p_2^v$ after decision-making and advertise $\overline{r^u}=(\uparrow (uv)p_2^v, \downarrow p^u)$ to neighbor node again. The theorem is true.

(4) If $\lambda^u((uv)p_1^v) > \lambda^u(p^u)$ and $\lambda^u((uv)p_2^v) > \lambda^u(p^u)$, the current optimal path of u is $s_t^u = (uv)p_2^v$ at time t. After receiving $r^v=(\uparrow p_1^v, \downarrow p_2^v)$, u should update optimal path to $(uv)p_1^v$ and advertise $r^u=(\uparrow (uv)p_1^v, \downarrow (uv)p_2^v))$ to neighbor nodes. At time $t+1$, the

current optimal path of u is $s_{t+1}^u=(uv)\,p_1^v$ and v advertises $\overline{r^v}=(\uparrow p_2^v,\downarrow p_1^v)$ again. As shown in Fig.2(d), u should update optimal path to $(uv)\,p_2^v$ after decision-making and advertise $\overline{r^u}=(\uparrow(uv)\,p_2^v,\downarrow(uv)\,p_1^v)$ to neighbor node again. The theorem is true.

The theorem has been proved.

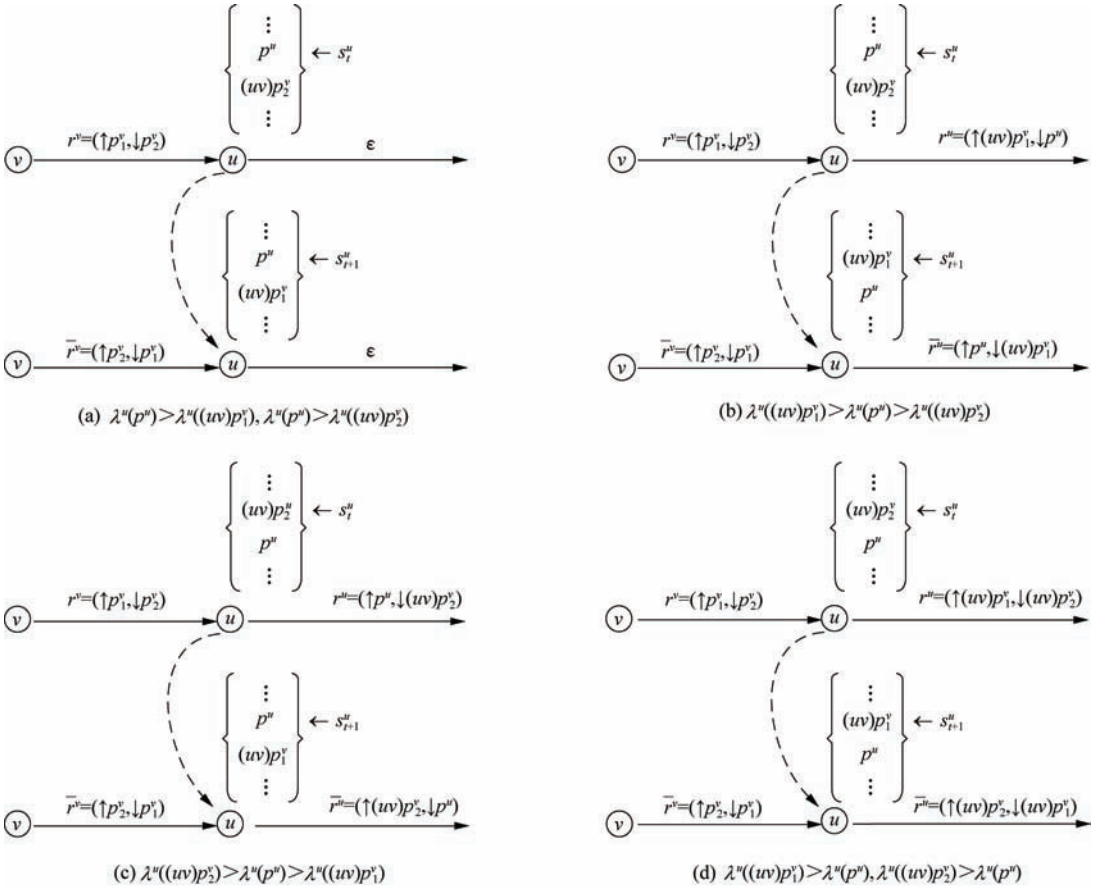

Fig.2 Response output of u after receiving consecutive positive-negative report (r^v and $\overline{r^v}$).

Definition 5 Given $G=(V, E)$ in which routing policies are stable at each node, there is a series of neighbor nodes $u,v,\cdots,w,\cdots,x\in V$, as shown in Fig.3. Sometimes update report r^v of v results in the fact that u advertises r^u, which will bring about a series of report output in turn, such as r^w and r^x. This series of report outputs is called route update chain, denoted as $r^v\to r^u\to\cdots\to r^w\to\cdots\to r^x$.

Fig.3 Route update chain.

Lemma 1 Suppose that route policies of all nodes are stable and have no conflict. If the update report output r^v results in $r^v\to r^u\to\cdots\to r^x$ at time t, the negative report output $\overline{r^v}$ will result in $\overline{r^v}\to\overline{r^u}\to\cdots\to\overline{r^x}$ at the next time $t+1$.

Proof According to the number of transmission edges of every node, there are two cases as follows:

(1) If maximum number of transmission edge is 2 as shown in Fig.4(a), which means each node has transmission edges with two neighbor nodes at most, this lemma is true according to Theorem 1.

(2) If the number of transmission edges is more than 2 at some nodes, for example z and y as shown in Fig.4(b), we assume that path information carried by r^z

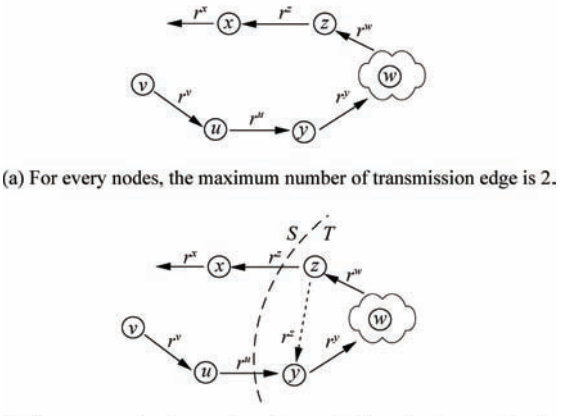

Fig.4 Examples of route update chain with different number of transmission edge at each node.

is accepted by Adj-RIB-In of y. Assume that (S,T) is a cut which partitions all nodes in route update chain into T and S, where T comprises those nodes from y to z and e_{xz}/e_{uy} are transmission edges crossing cut (S,T). Because route policies of all nodes are stable and have no conflict, T should have a stable route tree after receiving r^u at time t. Then T will advertise r^T to node x, which means $r^u \rightarrow r^T$. On the contrary, if receiving $\overline{r^u}$ at time t+1, i.e. r^u is canceled, T should recover former route status and cancel r^T too, which means $\overline{r^u} \rightarrow \overline{r^T}$. This lemma is also true according to Theorem 1.

The lemma has been proved.

Theorem 2 Given $G=(V, E)$ in which routing policies are stable at each node, there is a series of neighbor nodes $u,v,\cdots, x \in V$. Suppose that these nodes have no conflict with route policies except v. At time t, the route update report r^v results in route update chain $r^v \rightarrow r^u \rightarrow \cdots \rightarrow r^x$. If 1) v is a neighbor node of x too, and the direction of edge e_{vx} is $x \rightarrow v$, 2) the last-of-chain r^x also results in the fact that v advertises another negative report $\overline{r^v}$, i.e. $r^x \rightarrow \overline{r^v}$, route update chain will loops, which is composed of $r^v \rightarrow r^u \rightarrow \cdots \rightarrow r^x$ and $\overline{r^v} \rightarrow \overline{r^u} \rightarrow \cdots \rightarrow \overline{r^x}$. Along this loop chain, the same update report will appear repeatedly at v, which means route oscillation happens to v.

Proof As shown in Fig.5, the solid line represents route update chain $r^v \rightarrow r^u \rightarrow \cdots \rightarrow r^x$, and the broken line represents negative router update chain $\overline{r^v} \rightarrow \overline{r^u} \rightarrow \cdots \rightarrow \overline{r^x}$.

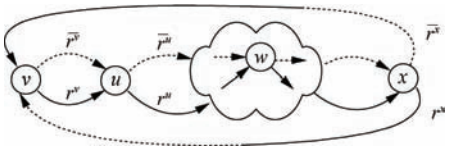

Fig.5 Route update loop chain.

If $r^x \rightarrow \overline{r^v}$, then $\overline{r^x} \rightarrow r^v$ according to Theorem 1.

r^x is not only the tail of chain $r^v \rightarrow r^u \rightarrow \cdots \rightarrow r^x$ but also the head of chain $\overline{r^v} \rightarrow \overline{r^u} \rightarrow \cdots \rightarrow \overline{r^x}$, contrary to $\overline{r^x}$. So $r^v \rightarrow r^u \rightarrow \cdots \rightarrow r^x$ and $\overline{r^v} \rightarrow \overline{r^u} \rightarrow \cdots \rightarrow \overline{r^x}$ will form a loop chain. Then route update report will transfer forwards in terms of

$$r^v \rightarrow \cdots \rightarrow r^x \rightarrow \overline{r^v} \rightarrow \cdots \rightarrow \overline{r^x} \rightarrow r^v \rightarrow \cdots \rightarrow r^x \rightarrow \overline{r^v} \rightarrow \cdots$$

It is clear that r^v will appear repeatedly at node v, which means route oscillation happens.

The theorem has been proved.

Lemma 2 For $G=(V, E)$, $u,v,\cdots,x \in V$, suppose that route policies of all nodes are stable and have no conflict except v. At time t, node v advertises route update report r^v through output transmission edge. If r^v results in the fact that v will receive another report r^x by which v should advertise $\overline{r^v}$ again, viz. consecutive reports $(r^v, \overline{r^v})$ is detected at node v, it is assured that route oscillation will occur at node v.

Proof If $r^v \rightarrow r^x$ and $r^x \rightarrow \overline{r^v}$ at time t, the chain $\overline{r^v} \rightarrow \overline{r^x}$ will be generated at time t+1 according to Lemma 1. According to Theorem 2, $r^v \rightarrow r^x$ and $\overline{r^v} \rightarrow \overline{r^x}$ will form a loop chain, which means route oscillation will occur at node v.

The lemma has been proved.

According to Lemma 2, a sufficient condition of route oscillation is that node detects both loop chain and consecutive positive-negative report in local record, while detailed routing policies and path information of other nodes are not required.

3. Design and Implementation

3.1. Approach design

For $G=(V, E)$, $u,v,\cdots, x \in V$, suppose that route policies of all nodes are stable and have no conflict except v. According to Lemma 2, if 1) consecutive positive-negative report $(r^v, \overline{r^v})$ is detected at node v, 2) r^v and $\overline{r^v}$ are involved in the same route update chain, it is assured that route oscillation will occur at node v.

Therefore, we present report update chain tag (RUCT) to judge whether r^v and $\overline{r^v}$ are involved in the same route update chain. RUCT is a network ID set of nodes which are traversed by forwarding update report. When network ID of one node has been included by RUCT, it is affirmable that this node should have been involved in this route update chain. So this chain should loop if this node advertises update report again. In addition, we make local library to record changing history of route update report, by which the consecutive positive-negative reports can be detected. For an arbitrary node, such as v, we make rules as follows:

(1) When receiving route update report r^x, v will advertise update report r^v and insert r^v into local library if the current optimal path is changed after route decision-making. If RUCT is not accompanied by r^x, v has to create RUCT and add local network ID to RUCT. Then v will advertise r^v together with RUCT to other neighbor nodes.

(2) After receiving r^v together with RUCT, neighbor node firstly makes decision to check whether the optimal path will be changed. If there is no change, it is unnecessary to transfer forwards RUCT, which means that RUCT should end here. Otherwise, this node should add its network ID to RUCT and revise local record. Then this node should transfer forwards RUCT together with the new update report.

(3) If receiving update report again, v should judge whether the optimal path will be changed or not. If it is not changed, v stops forwarding RUCT similarly. Oth-

erwise, v should get the new update report $r^{v'}$ firstly, and check 1) whether $r^{v'}$ is equal to $\overline{r^{v}}$ or not. If yes, the consecutive positive-negative report is detected. If no, v should refresh $r^{v'}$ into local history library; 2) whether the received RUCT includes network ID of v or not. If yes, r^{v} and $r^{v'}$ have already been involved in the same chain. Otherwise, v should add local network ID to RUCT. Only if both 1) and 2) are true, can we conclude that route oscillation will occur at node v. Otherwise, v should forward $r^{v'}$ together with RUCT.

According to the above rules, each node should only add local network ID to RUCT rather than share detailed routing policies or path information. Whether route oscillation would occur at each node can be judged by checking RUCT and local history record, which improves the security and secrecy of information.

3.2. Extending BGP

RUCT consists of network IDs, and we can extend BGP to construct and transfer forwards RUCT. The additional routing parameter of BGP can be assigned to a special path attribute of route update report. A path attribute consists of one byte of flag, one byte of code, one or two bytes of length and multiple bytes of data. Then we extend an optional non-transitive path attribute to express RUCT, named PRUCT.

As shown in Fig.6, attribute code of PRUCT is 31 (1F in hexadecimal), attribute length of PRUCT is two bytes (which is enough to be fit for most applications), and attribute data of PRUCT is loaded by a series of network IDs.

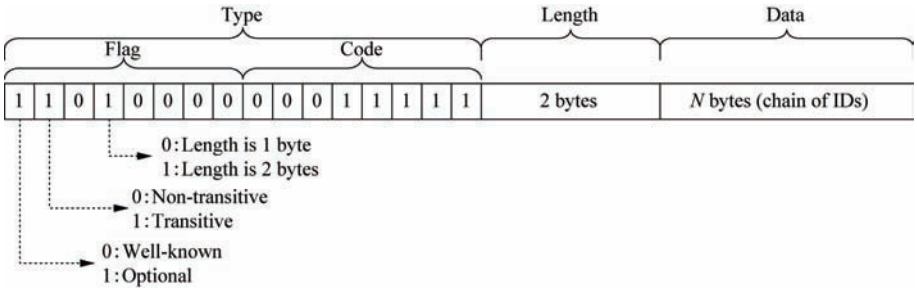

Fig.6 Format for PRUCT.

3.3. Pseudo code

We design the pseudo code for each node to implement approach based on RUCT, shown in Fig.7.

```
if best_change(u) then {
    r_new = getReport(u)
    if receive_ruct() then {
        RUCT = receive_ruct();
        if RUCT.find(getID(u)) then {
            if r_h = negative(r_new) then {
                do_oscillation(u);
            }
            else {
                r_h = r_new
                for each v ∈ peers(u)
                    send (r_new, RUCT) to v
            }
        }
        else {
            RUCT.add(getID(u))
            for each v ∈ peers(u)
                send (r_new, RUCT) to v
        }
    }
    else {
        RUCT = create_ruct(u)
        RUCT.add(getID(u))
        for each v ∈ peers(u)
            send (r_new, RUCT) to v
    }
}
```

Fig.7 Pseudo code.

For an arbitrary node u, maybe the optimal path changes when update report is received or local routing policies are modified. Firstly, we get the new route update report r_{new} and detect whether RUCT has been carried or not. If not, it means that the changing of optimal path is due to modification of local routing policies or network topology. Then RUCT should be created and transferred forwards together with r_{new}. If RUCT is carried, we will go on to check whether RUCT includes local network ID or not. If RUCT does not include local ID, it is necessary to add local ID to RUCT and transfer forwards RUCT together with r_{new}. And the history record r_h should be updated with r_{new}. If RUCT includes local ID, it means that the update report chain loops. Then we will go on to check whether r_h is equal to $\overline{r_{new}}$ or not. If $r_h=\overline{r_{new}}$, it is assured that route oscillation should happen according to Lemma 2. Otherwise, r_h should be updated with r_{new}, and RUCT should be transferred forwards without being modified.

4. Simulation Experiment

We demonstrate this approach on traditional model. As shown in Fig.8, this model consists of four nodes. Each node represents a subnet, and Node 0 means destination. The numeral arrays beside nodes are all feasible paths, which array by priority from the top down.

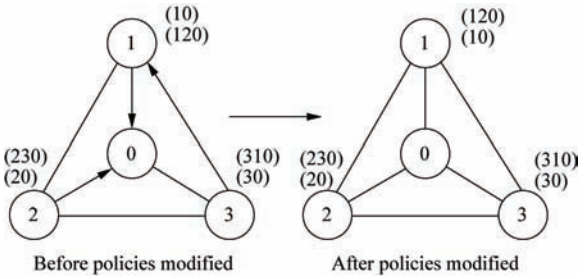

Fig.8 Route oscillation model.

We simulate this model by C language, and the result is shown in Fig.9.

Initially the whole network has a stable routing tree ((10), (20), (310)), which is tagged by arrow as shown in Fig.8. Sometimes, the routing policies of Node 1 are modified, where the priority of path (120) is adjusted to be higher than priority of path (10). Then route oscillation will happen.

At time 0, the whole network has a stable routing tree.

At time 1, the policies of Node 1 are modified, where the optimal path is changed from (10) to (120). Node 1 will create update report r^1=(↑(120),↓(10)) and RUCT={1}, and will transfer them forwards to other nodes. In addition, the history record of Node 1 will be updated with r_h^1=(↑(120),↓(10)).

At time 2, Node 2 does not respond after receiving

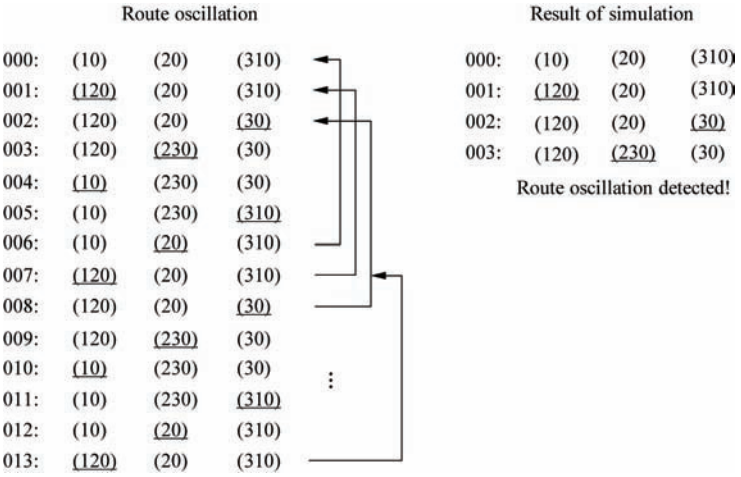

Fig.9 Simulation result (the path with underline means that owner node should advertise update report).

update report. On the other hand, Node 3 will change optimal path after decision-making. It is obviously that RUCT={1} does not include network ID of Node 3 and the history record of Node 3 is empty. Node 3 should change RUCT={1} to RUCT={1,3} and create update report r^3=(↑(30),↓(310)), then transfer them forwards. The history record of Node 3 will be updated with r^3=(↑(30),↓(310)).

At time 3, in the same way, Node 2 will change RUCT={1,3} to RUCT={1,3,2} and create r^2=(↑(230),↓(20)), then transfer them forwards to other neighbor nodes. Similarly, the local record of Node 2 will be updated with r_h^2=(↑(230),↓(20)).

At time 4, Node 1 receives r^2=(↑(230),↓(20)) from Node 2. Then Node 1 should change optimal path and create new update report r^1=(↑(10),↓(120)) which is equal to negative report of history record r_h^1=(↑(120), ↓(10)), i.e. $\overline{r_h^1}$. In addition, network ID of Node 1 has been included in the received RUCT= {1,3,2}, which means the update report chain loops. Therefore we can assure that route oscillation will occur at Node 1.

We compare this approach based on RUCT (signed by RUCT) with approach based on relative preference[21] (signed by RP) and approach based on TOKEN[22] (signed by TOKEN). Both RP and TOKEN respect privacy of routing policies and have optimal ultimate route tree after eliminating oscillation, which means they are typically similar researches with RUCT. Figs.10-11 show the comparison of on checking time and cost between three approaches. The results demonstrate that RUCT is available and efficient.

Fig.10 shows the comparison result of checking time whose unit is one MRAI. The checking time of RP is the largest among three approaches, which is more than triple of TOKEN's when MAX_TIMES of RP (upper limit of conflicts) is 4. And the checking time of RUCT

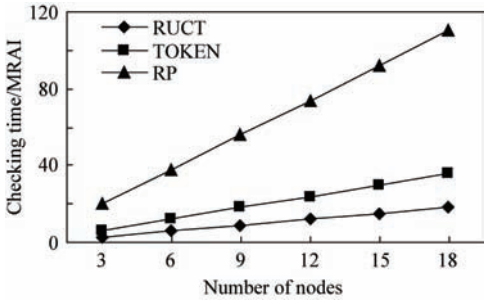

Fig.10 Comparison of checking time.

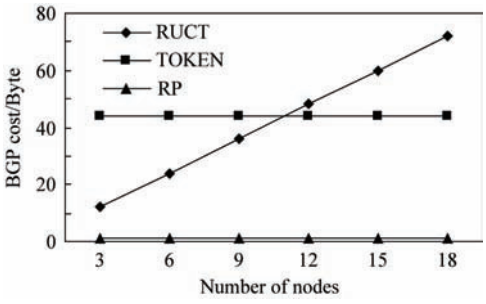

Fig.11 Comparison of BGP cost.

is the smallest, which is half of TOKEN's.

Fig.11 shows the comparison result of BGP cost. The cost of RP is the least among three approaches, which spends constantly one byte, viz. relative preference. The cost of TOKEN is constant too, which is occupied by attribute flag of node and conflict path after being encrypted. The cost of RUCT is an incremental variable about number of nodes, which is less than cost of TOKEN if the number is small and should exceed cost of TOKEN if the number is too large. However, the total BGP cost of RUCT is very finite because network ID spends merely few bytes. Therefore, RUCT cannot markedly aggravate network cost of BGP.

One thing to be noted is that RUCT requires no additional transmission cost because it is merely a path attribute carried by BGP update report. But the number of nodes included in RUCT is restrained by the maximum message size of BGP update report. As is known to all, the maximum message size of BGP update report is 4 096 bytes, which covers 19-byte BGP head and other necessary properties. So the number of nodes included in RUCT is actually less than 1 000 when network ID of each node occupies 4 bytes.

5. Conclusions

(1) The RUCT based approach can greatly respect the privacy of routing policies of AS when checking route oscillation.

(2) Compared with other similar approaches, the proposed approach spends the smallest checking time by equivalent network cost of BGP.

(3) This approach only focuses on consecutive positive-negative report in local record. The multi-phase state conversion of local reports will be possibly studied in future.

References

[1] Rekhter Y, Li T, Hares S. A border gateway protocol (BGP-4). RFC 4271, 2006.

[2] Butler K, Farley T R, McDaniel P, et al. A survey of BGP security issues and solutions. Proceedings of the IEEE 2010; 98(1): 100-122.

[3] Wang F, Qiu J, Gao L X, et al. On understanding transient interdomain routing failures. IEEE/ACM Transactions on Networking 2009; 17(3): 740-751.

[4] Oliveira R, Zhang B C, Pei D, et al. Quantifying path exploration in the Internet. IEEE/ACM Transactions on Networking 2009; 17(2): 445-458.

[5] Afek Y, Bremler-Barr A, Schwarz S. Improved BGP convergence via ghost flushing. IEEE Journal on Selected Areas in Communications, 2004; 22(10): 1933-1948.

[6] Nicholes M, Mukherjee B. A survey of security techniques for the border gateway protocol (BGP). IEEE Communications Surveys & Tutorial 2009; 11(1): 52-65.

[7] Ortiz S. Securing the Internet's routing infrastructure. Computer 2009; 42(4): 21-23.

[8] Kuhn R, Liu S, Rossman H. Practical interdomain routing security. IT Professional 2009; 11(6): 54-56.

[9] Govindan R, Alaettinoglu C, Eddy G, et al. An architecture for stable, analyzable Internet routing. IEEE Network 1999; 13(1): 29-35.

[10] Griffin T G, Wilfong G. An analysis of BGP convergence properties. Proceedings of the ACM SIGCOMM. 1999; 214-223.

[11] Zhao H Q, Sun J, Gao Y, et al. An improved project for testing conflicting policies of BGP routing. Journal of China Institute of Communications, 2002; 23(7): 91-97. [in Chinese]

[12] Zhao H Q, Sun J. A study of algorithm for testing route oscillation based on algebraic method. Chinese Journal of Computers 2007; 30(10): 1763-1769. [in Chinese]

[13] Villamizar C, Chandra R, Govindan R. BGP route flap damping. RFC 2439, 1998.

[14] Mao Z M, Govindan R, Varghese G, et al. Route flap damping exacerbates Internet routing convergence. Proceedings of the ACM SIGCOMM. 2002; 221-233.

[15] Griffin T G, Shepherd F B, Wilfong G. The stable paths problem and interdomain routing. IEEE/ACM Transactions on Networking, 2002; 10(2): 232-243.

[16] Ee C T, Ramachandran V, Chun B G, et al. Resolving BGP disputes. University of California, Berkeley: Technical Reports, EECS-2006-39, 2006.

[17] Griffin T G, Wilfong G. A safe path vector protocol. Proceedings of IEEE INFOCOM. 2000; 490-499.

[18] Murphy S. BGP security vulnerabilities analysis. RFC 4272, 2006.

[19] Murphy S L. Secure inter-domain routing standards evolution and role in the future GIG. IEEE Military Communications Conference. 2007; 1-7.

[20] Budenske J R, Millikin K S, Bonney J C, et al. Space network architecture technologies. IEEE/Aerospace Conference Proceedings. 2002; 3: 1061-1069.

[21] Wang H J, Wang D D, Liang H Y, et al. Eliminating BGP oscillations caused by policy conflicts. IEEE International Symposium on Communications and Information Technology. 2005; 201-204.

[22] Ahronovitz E, Konig J C, Saad C. A distributed method for dynamic resolution of BGP oscillations. 20th International Parallel and Distributed Processing Symposium. 2006.

[23] Ee C T, Ramachandran V, Chun B G, et al. Resolving inter-domain policy disputes. Proceedings of the ACM SIGCOMM. 2007; 157-168.

[24] Hu Z Y, Zhang J. A novel approach for checking route oscillation of BGP. IEEE/AIAA28th Digital Avionics Systems Conference. 2009; 7.B.4-1-7.B.4-6.